ANSI X3.23a-1989

ADOPTED FOR USE BY
THE FEDERAL GOVERNMENT

FIPS

PUB 21-3

SEE NOTICE ON INSIDE

American National Standard

for Information Systems –

Programming Language – Intrinsic Function Module for COBOL

ANSI X3.23a-1989

ANSI American National Standards Institute
1430 Broadway
New York , New York
10018

This standard has been adopted for Federal Government use.

Details concernings its use within the Federal Government are contained in Federal Information Processing Standards Publication 21-3, COBOL. For a complete list of the publications available in the Federal Information Processing Standards Series, write to the Standards Processing Coordinator (ADP), National Institute of Standards and Technology, Gaithersburg, MD 20899.

ANSI ®
X3.23a-1989
Supplement to
ANSI X3.23-1985

American National Standard
for Information Systems –

Programming Language –
Intrinsic Function Module for COBOL

Secretariat

Computer and Business Equipment Manufacturers Association

Approved September 13, 1989

American National Standards Institute, Inc

ACKNOWLEDGMENT

American National Standard

Approval of an American National Standard requires verification by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer.

Consensus is established when, in the judgment of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that a concerted effort be made toward their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether he has approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give an interpretation of any American National Standard. Moreover, no person shall have the right or authority to issue an interpretation of an American National Standard in the name of the American National Standards Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

CAUTION NOTICE: This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

Published by

American National Standards Institute
1430 Broadway, New York, New York 10018

Printed in the United States of America

APS15C590/32

Foreword

(This Foreword is not part of American National Standard X3.23a-1989.)

This supplement, the first to the document entitled "American National Standard for Information Systems – Programming Language – COBOL, ANSI X3.23-1985, ISO 1989-1985," presents a new COBOL module, the Intrinsic Function module. This module provides the capability of referencing a data item whose value is derived automatically during the execution of a program.

In June 1985, Technical Committee X3J4 on COBOL of Accredited Standards Committee X3 began the task of preparing supplements that will add upwardly mobile, compatible COBOL language extensions. The extensions that X3J4 considered were taken from proposals that appeared in *CODASYL COBOL Journal of Development*, 1984.

In January 1987, X3J4 approved the content and format for the first supplement and recommended to X3 that the proposed draft be published for public review and comment by the data-processing community of software producers and users. During two public review and comment periods, X3J4 reviewed and responded to all the comments.

In October 1988, X3J4 approved the final version of the proposed supplement and forwarded it to X3 for processing. X3 approved the proposed supplement and submitted it to the American National Standards Institute. This supplement was designated ANSI X3.23a-1989 and was approved by ANSI on September 13, 1989.

Suggestions for improvement of this standard will be welcome. They should be sent to the Computer and Business Equipment Manufacturers Association, 311 First Street, NW, Suite 500, Washington, DC 20001.

This standard was processed and approved for submittal to ANSI by the Accredited Standards Committee on Information Processing Systems, X3. Committee approval of the standard does not necessarily imply that all committee members voted for its approval. At the time it approved this standard, the X3 Committee had the following members:

Richard Gibson, Chair
Donald Loughry, Vice-Chair
(Vacant), Administrative Secretary

Organization Represented	*Name of Representative*
Allen-Bradley Company	Ronald H. Reimer
American Library Association	Paul Peters
American Nuclear Society	Geraldine C. Main
AMP, Inc	Edward Kelly
	Ronald Lloyd (Alt)
Apple	Karen Higginbottom
	Michael J. Lawler (Alt)
Association of the Institute for Certification of Computer Professionals	Thomas M. Kurihara
AT&T	Thomas F. Frost
	Paul D. Bartoli (Alt)
Boeing Company	Paul W. Mercer
Compaq Computer Corporation	James Barnes
Control Data Corporation	Ernest Fogle
Cooperating Users of Burroughs Equipment	Thomas Easterday
	Donald Miller (Alt)
Dataproducts Corporation	Charles D. Card
Digital Equipment Computer Users Society	James Ebright
Digital Equipment Corporation	Gary S. Robinson
	Delbert L. Shoemaker (Alt)

Organization Represented	*Name of Representative*
Eastman Kodak	Gary Haines
	James D. Converse (Alt)
Electronic Data Systems Corporation	Jerrold S. Foley
GUIDE International	Frank Kirshenbaum
	Jeffrey Roberts (Alt)
Hewlett-Packard	Donald C. Loughry
Honeywell Bull	David M. Taylor
IBM Corporation	Mary Anne Gray
	Robert H. Follett (Alt)
IEEE Computer Society	Tom Hannon
	Bob Pritchard (Alt)
Lawrence Berkeley Laboratory	David F. Stevens
	Robert L. Fink (Alt)
MAP/TOP	Mike Kaminski
Moore Business Forms	Delmer H. Oddy
National Communications System	Dennis Bodson
	Donald Wilson (Alt)
National Institute of Standards and Technology	Robert E. Rountree
	Mike Hogan (Alt)
NCR Corporation	Tom Kern
	A. R. Daniels (Alt)
OMNICOM	Harold C. Folts
	Cheryl Slobodian (Alt)
Prime Computer, Inc	Tom Connerty
	Phil Cieply (Alt)
Recognition Technology Users Association	Herbert F. Schantz
SHARE Inc	Thomas B. Steel
	Gary Ainsworth (Alt)
3M Company	Paul D. Jahnke
Unisys	Marvin W. Bass
	Steven Oksala (Alt)
U.S. Department of Defense	William C. Rinehuls
	Thomas H. Kurihara (Alt)
U.S. General Services Administration	Dale O. Christensen
	Larry L. Jackson (Alt)
US WEST	Gary Dempsey
	Sue Capraro (Alt)
VIM	Chris Tanner
	John Ulrich (Alt)
Wang Corporation	J. J. Cinecoe
	Sarah Wagner (Alt)
Wintergreen Information Services	John L. Wheeler
Xerox Corporation	Roy Pierce

Technical Committee X3J4 on COBOL, which developed this addendum, had the following members:

D.A. Schricker, Chair
B. D. Sinclair, Vice Chair
P. A. Beard, Secretary

P. Blacklock
W. Blatt
D. Bowman
J. R. Brieschke
V. Eckels
C. P. Ellis
A. Fryer
J. Garfunkel
B. Gaarder
P. Graham
A. Hewitt
K. Howard
A. Jackson
J. Kailey
R. Kisselburgh
S. D. Klute
P. L'Allier
L. K. Madison
D. D. Marriott
J. R. Peters
A. O. Reimann
J. A. Rodriguez
A. Saturnelli
L. Skidmore
S. Spears
S. Spence
W. Stover
M. V. Vickers
A. Wallace
K. Watts
L. V. Willis

Others who contributed to the work on the addendum were as follows:

M. Adams
M. Blanchette
P. Brown
M. Daleo
B. J. Edwards
P. B. Hall
J. M. Hart
C. Johnson
L. Keating
J. A. Moran
B. M. Strauss
J. M. Triance
J. P. Wessler

TABLE OF CONTENTS

American National Standard
for Information Systems –

Programming Language –
Intrinsic Function Module for COBOL

ADDENDUM 1: CHANGES TO STANDARD COBOL

The following are the changes to be applied to the document entitled "American National Standard for Information Systems - Programming Language - COBOL, ANSI X3.23-1985, ISO 1989-1985", in order to include the language elements of the Intrinsic Function module and to correct typographical errors (indicated by the symbols (T) in the page number column). The changes add totally upward compatible language extensions to Standard COBOL except for the addition of the reserved word FUNCTION.

Page No.	Change to ANSI X3.23-1985 and ISO 1989-1985
ii	Add the following after the entry for section II, paragraph 7:
	8. Intrinsic Function Facility II-36
iii	Change the entry for section VI, paragraph 1.3 to read:
	1.3 Restrictions on Overall Language VI-1
xi	Add the following after the entry for 3.4:
	3.5 American National Standard COBOL 1985, Addendum 1 . . . . . XVII-13
xi	Add the following after the entry for 4.3:
	4.4 ISO Standard 1989-1985 for COBOL, Addendum 1 . . . . . XVII-15
I-1	Paragraph 1.2, first paragraph, second sentence, change to read: "The standard defines 12 functional processing modules: Nucleus, Sequential I-O, Relative I-O, Indexed I-O, Inter-Program Communication, Sort-Merge, Source Text Manipulation, Report Writer, Communication, Debug, Segmentation, and Intrinsic Function."
I-1	Paragraph 1.2, first paragraph, last sentence, change to read: "Three of the modules contain only level 1 elements."
I-3	Paragraph 1.2, add the following paragraph after the last paragraph:
	The Intrinsic Function module provides the capability to reference a data item whose value is derived automatically at the time of reference during the execution of the object program. The Intrinsic Function module contains only level 1 elements.

Page No.	Change to ANSI X3.23-1985 and ISO 1989-1985
I-3	Paragraph 1.3, first paragraph, first sentence, change to read: "...into nineteen sections."
I-3	Paragraph 1.3, third paragraph, first sentence, change to read: "Sections VI through XVI and section A contain specifications for the twelve functional...".
I-3	Paragraph 1.3, fourth paragraph, change to read: "Sections II through XVI and section A comprise the detailed...".
I-4	Paragraph 1.4, fourth paragraph, first sentence, change to read: "...representation of the 12 functional processing modules...".
I-4	Paragraph 1.4, fifth paragraph, add to the list: ITR Intrinsic Function
I-5	Change diagram to include the Intrinsic Function module as shown on the next page of this addendum.

		REQUIRED MODULES (Required in Subsets)							OPTIONAL MODULES (Not Required in Subsets)				
		Nucleus	Sequential I-O	Relative I-O	Indexed I-O	Inter-Program Communication	Sort-Merge	Source Text Manipulation	Report Writer	Communication	Debug	Segmentation	Intrinsic Function
COBOL SUBSETS	HIGH	2 NUC 1,2	2 SEQ 1,2	2 REL 0,2	2 INX 0,2	2 IPC 1,2	1 SRT 0,1	2 STM 0,2	1 RPW 0,1	2 COM 0,2	2 DEB 0,2	2 SEG 0,2	1 ITR 0,1
	INTERMEDIATE	1 NUC 1,2	1 SEQ 1,2	1 REL 0,2	1 INX 0,2	1 IPC 1,2	1 SRT 0,1	1 STM 0,2					
	MINIMUM	1 NUC 1,2	1 SEQ 1,2	NULL	NULL	1 IPC 1,2	NULL	NULL		1 COM 0,2	1 DEB 0,2	1 SEG 0,2	

Page No.	Change to ANSI X3.23-1985 and ISO 1989-1985

I-6 Paragraph 1.5, second sentence, change "consists of 11 modules, seven of which are required and four of which are optional." to "consists of 12 modules, seven of which are required and five of which are optional. (It is the intention of X3J4 to require the Intrinsic Function module in the next complete revision of Standard COBOL.)"

I-6 Paragraph 1.5.1, first paragraph, last sentence, change to read: "The five optional modules (Report Writer, Communication, Debug, Segmentation, and Intrinsic Function) are not...".

I-8 Paragraph 1.5.2.5.3, change to read:

1.5.2.5.3 Reserved Words

An implementation of Standard COBOL must recognize as reserved words all of the COBOL reserved words occurring in the specification of the seven required modules and the four optional modules of Report Writer, Communication, Debug, and Segmentation. An implementation of Standard COBOL need not recognize any new reserved words introduced by the optional Intrinsic Function module until that module is included in the implementation. (See page IV-45, COBOL Reserved Words.)

I-10 Paragraph 2.1, last paragraph, add the following to the list:

- Page I-39: Summary of elements in the Intrinsic Function module

I-12 Change the entry for "Reference modification" to "Reference-modifier".

I-12 (T) Under "Reference Format", change "Asterisk (8) comment line" to "Asterisk (*) comment line".

I-39 After the Segmentation module list, add the Intrinsic Function module list shown on the next page of this addendum.

SUMMARY OF ELEMENTS IN THE INTRINSIC FUNCTION MODULE

ELEMENT	LEVEL 1
LANGUAGE CONCEPTS	
Character-Strings	
COBOL words	
Function-name	X
Uniqueness of Reference	
Function-Identifier	X
PROCEDURE DIVISION	
ACOS function	X
ANNUITY function	X
ASIN function	X
ATAN function	X
CHAR function	X
COS function	X
CURRENT-DATE function	X
DATE-OF-INTEGER function	X
DAY-OF-INTEGER function	X
FACTORIAL function	X
INTEGER function	X
INTEGER-OF-DATE function	X
INTEGER-OF-DAY function	X
INTEGER-PART function	X
LENGTH function	X
LOG function	X
LOG10 function	X
LOWER-CASE function	X
MAX function	X
MEAN function	X
MEDIAN function	X
MIDRANGE function	X
MIN function	X
MOD function	X
NUMVAL function	X
NUMVAL-C function	X
ORD function	X
ORD-MAX function	X
ORD-MIN function	X
PRESENT-VALUE function	X
RANDOM function	X
RANGE function	X
REM function	X
REVERSE function	X
SIN function	X
SQRT function	X
STANDARD-DEVIATION function	X
SUM function	X
TAN function	X
UPPER-CASE function	X
VARIANCE function	X
WHEN-COMPILED function	X

Page No.	Change to ANSI X3.23-1985 and ISO 1989-1985
I-40	Paragraph 3.1, second paragraph, add the following to the list: ITR Intrinsic Function
I-41 (T)	Under "Character Set", change entry in MODULE column for "Characters used in punctuation =" from "1 STM" to "2 STM".
I-42	Add the following before the entry for "Literals"; align the word "Function-names" one position to the right of the word "Literals": Function-names . . . . . 1 ITR
I-42	Change the entry for "Reference modification" to "Reference-modifier".
I-42	Add the following before the entry for "Reference-modifier"; align the word "Function-identifier" with the word "Reference-modifier": Function-identifier . . . . . 1 ITR
I-45 (T)	Under "OBJECT-COMPUTER paragraph", change entry in MODULE column for "SEGMENT-LIMIT clause" from "1 SEG Z" to "2 SEG Z".
I-51 (T)	In the line following "Level-number clause", change "may be 1 or 1 digits" to "may be 1 or 2 digits".
I-54	Add the following before the entry for "ACCEPT statement": Intrinsic functions ACOS function . . . . . 1 ITR ANNUITY function . . . . . 1 ITR ASIN function . . . . . 1 ITR ATAN function . . . . . 1 ITR CHAR function . . . . . 1 ITR COS function . . . . . 1 ITR CURRENT-DATE function . . . . . 1 ITR DATE-OF-INTEGER function . . . . . 1 ITR DAY-OF-INTEGER function . . . . . 1 ITR FACTORIAL function . . . . . 1 ITR INTEGER function . . . . . 1 ITR INTEGER-OF-DATE function . . . . . 1 ITR INTEGER-OF-DAY function . . . . . 1 ITR INTEGER-PART function . . . . . 1 ITR LENGTH function . . . . . 1 ITR LOG function . . . . . 1 ITR LOG10 function . . . . . 1 ITR LOWER-CASE function . . . . . 1 ITR MAX function . . . . . 1 ITR MEAN function . . . . . 1 ITR MEDIAN function . . . . . 1 ITR MIDRANGE function . . . . . 1 ITR MIN function . . . . . 1 ITR MOD function . . . . . 1 ITR

NUMVAL function . . . . . 1 ITR
NUMVAL-C function . . . . . 1 ITR
ORD function . . . . . 1 ITR
ORD-MAX function . . . . . 1 ITR
ORD-MIN function . . . . . 1 ITR
PRESENT-VALUE function . . . . . 1 ITR
RANDOM function . . . . . 1 ITR
RANGE function . . . . . 1 ITR
REM function . . . . . 1 ITR
REVERSE function . . . . . 1 ITR
SIN function . . . . . 1 ITR
SQRT function . . . . . 1 ITR
STANDARD-DEVIATION function . . . . . 1 ITR
SUM function . . . . . 1 ITR
TAN function . . . . . 1 ITR
UPPER-CASE function . . . . . 1 ITR
VARIANCE function . . . . . 1 ITR
WHEN-COMPILED function . . . . . 1 ITR
Statements

I-54 thru I-63 Indent all lines following the newly inserted "Statements" entry beginning with "ACCEPT statement" and ending with "END-WRITE phrase".

II-23 (T) Paragraph 6.4.1.1, entitled "Names of Programs", second paragraph, first line, change "compiled program" to "compiled programs".

II-25 (T) Paragraph 6.4.2.2, entitled "Value of Parameters", second paragraph, penultimate line, change "may be used by a called program to return to the" to "may be used by a called program to return a result to the".

II-35 Add the following on page II-36 after paragraph 7.6:

8. INTRINSIC FUNCTION FACILITY

Data processing problems frequently require the use of values which are not directly accessible in the data storage associated with the object program. These data values must be derived through performing some operations on other data. A function represents a temporary data item whose value is derived automatically at the time of reference during the execution of the object program.

The value returned by a function is considered to be a data value. A mechanism is provided at object time to assign a data value to a function when it is referenced. In order to determine the value of a function, the evaluation mechanism may require access to data values provided by the referencing program. These data values are provided by specifying parameters, known as arguments, when referencing the function. Specific functions may place constraints on these arguments such as range, data types, or size, etc. If, at the time a function is referenced, the arguments specified for that reference do not have values that comply with the specified constraints, the returned value for the function is undefined.

III-1 thru III-26

Insert the following terms into the glossary at the appropriate position in the alphabetic sequence:

Alphanumeric Function. A function whose value is composed of a string of one or more characters from the computer's character set.

Argument. An identifier, a literal, or an arithmetic expression that specifies a value to be used in the evaluation of a function.

Function. A temporary data item whose value is determined by invoking a mechanism provided by the implementor at the time the function is referenced during the execution of a statement.

Function-Identifier. A syntactically correct combination of character-strings and separators that references a function. The data item represented by a function is uniquely identified by a function-name with its arguments, if any. A function-identifier may include a reference-modifier. A function-identifier that references an alphanumeric function may be specified anywhere in the general formats that an identifier may be specified, subject to certain restrictions. A function-identifier that references an integer or numeric function may be referenced anywhere in the general formats that an arithmetic expression may be specified. (See page IV-22, Function-Identifier.)

Function-Name. A word that names a mechanism provided by the implementor to determine the value of a function.

Integer Function. A function whose category is numeric and whose definition provides that all digits to the right of the decimal point are zero in the returned value for any possible evaluation of the function.

Numeric Function. A function whose class and category are numeric but which for some possible evaluation does not satisfy the requirements of an integer function.

III-1 thru III-26

Replace the definitions as follows:

Data Item. A unit of data (excluding literals) defined by a COBOL program or by the rules for function evaluation.

Identifier. A syntactically correct combination of character-strings and separators that names a data item. When referencing a data item which is not a function, an identifier consists of a data-name, together with its qualifiers, subscripts, and reference-modifier, as required for uniqueness of reference. When referencing a data item which is a function, a function-identifier is used. The rules for 'identifier' associated with general formats may, however, specifically prohibit reference to functions, qualification, subscripting, or reference modification.

Integer. (1) A numeric literal that does not include any digit positions to the right of the decimal point.

(2) A numeric data item defined in the Data Division that does not include any digit positions to the right of the decimal point.

(3) A numeric function whose definition provides that all digits to the right of the decimal point are zero in the returned value for any possible evaluation of the function.

Where the term 'integer' appears in the general formats, integer must be a numeric literal which is an integer, and it must be neither signed nor zero unless explicitly allowed by the rules for that format.

Key Word. A reserved word or function-name whose presence is required when the format in which the word appears is used in a source program.

Subscript. An occurrence number represented by either an integer, a data-name optionally followed by an integer with the operator + or -, or an index-name optionally followed by an integer with the operator + or -, that identifies a particular element in a table. A subscript may be the word ALL when the subscripted identifier is used as a function argument. (See page A-28, Arguments.)

Word. A character-string of not more than 30 characters which forms a user-defined word, a system-name, a reserved word, or a function-name. (See page IV-5, COBOL Words.)

III-19 Replace the definition for Reference Modifier with the following:

Reference-Modifier. A syntactically correct combination of character-strings and separators that defines a unique data item. It includes a delimiting left parenthesis separator, the leftmost character position, a colon separator, optionally a length, and a delimiting right parenthesis separator. (See page IV-22, Reference-Modifier.)

IV-2 (T) Paragraph 2.1.5 entitled "Ellipses", second paragraph, first line, change "In the general format," to "In the general formats,".

IV-4 (T) Paragraph 4.1, entitled "Character Set", third paragraph, first line, change "fewer than 51 characters, double" to "fewer than 52 characters (all characters of the COBOL character set except the lowercase letters), double".

IV-4 Paragraph 4.2.1, rule (4), change second sentence to read: "Parentheses may appear only in balanced pairs of left and right parentheses delimiting subscripts, a list of function arguments, reference modifiers, arithmetic expressions, or conditions."

IV-5 Paragraph 4.2.2.1, replace with the following:

A COBOL word is a character-string of not more than 30 characters which forms a user-defined word, a system-name, a reserved word, or a function-name. Each character of a COBOL word is selected from the set of letters, digits, and the hyphen. The hyphen may not appear as the first or last character. Each lowercase letter is considered to be equivalent to its corresponding uppercase letter. Within a source program, reserved words and user-defined words form disjoint sets; reserved words and system-names form disjoint sets; reserved words and function-names form disjoint sets; function-names, system-names, and user-defined words form intersecting sets. The same COBOL word may be used as a function-name, as a system-name, and as a user-defined word within a source program; and the class of a specific occurrence of the COBOL word is determined by the context of the clause or phrase in which it occurs.

IV-9 Add the following before paragraph 4.2.2.2:

4.2.2.1.4 Function-Names

A function-name is a word that is one of a specified list of words which may be used in COBOL source programs. The same word, in a different context, may appear in a program as a user-defined word or a system-name. (See page A-29, Definitions of Functions.)

IV-15 Paragraph 4.3.3, delete the last sentence and add the following new paragraphs:

Every data item which is a function is an elementary item, and belongs to one of the categories alphanumeric or numeric, and to the corresponding class; the category of each function is determined by the definition of the function. This definition is made in these specifications. (See page A-27, Intrinsic Function Module.)

The following table depicts the relationship of the class and categories of data items.

IV-16 Paragraph 4.3.4, second paragraph, replace first sentence with the following: "When a computer provides more than one means of representing data, the standard data format must be used for data items other than integer and numeric functions, if not otherwise specified by the data description."

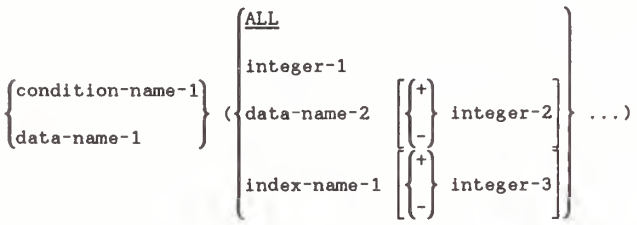
IV-16 Paragraph 4.3.4, add the following new paragraph preceding the last paragraph:

An alphanumeric function is always represented in the standard data format. The size of an alphanumeric function in standard data format characters is determined by the definition of the function. The implementor specifies the representation of integer and numeric functions, and this representation need not be the standard data format. Integer and numeric functions may be used only in arithmetic expressions. An integer or numeric function represents the value resulting from the evaluation of the function without the restriction on composite of operands and/or receiving data items. Each implementor will indicate the techniques used in evaluating integer and numeric functions.

IV-18 Paragraph 4.3.8, add the following new paragraph preceding the last paragraph:

When the resource named is a function, the function definition may require the user to specify in the reference to the function a value or set of values for one or more parameters which determine the value of the function for that particular reference. This is accomplished through the specification of arguments as described in the following paragraphs.

IV-21 Paragraph 4.3.8.2.2, replace the general format with the following:



IV-21 Paragraph 4.3.8.2.3, add new syntax rule 7 as follows:

(7) The subscript ALL may be used only when the subscripted identifier is used as a function argument and may not be used when condition-name-1 is specified. (See page A-28, Arguments.)

IV-22 thru IV-24 Renumber paragraphs 4.3.8.3 thru 4.3.8.5 to 4.3.8.4 thru 4.3.8.6, respectively.

IV-22 Add the following as new paragraph 4.3.8.3:

4.3.8.3 Function-Identifier

4.3.8.3.1 Purpose of a Function-Identifier

A function-identifier is a syntactically correct combination of character-strings and separators that uniquely references the data item resulting from the evaluation of a function.

4.3.8.3.2 General Format

```
FUNCTION function-name-1 [({argument-1} ... )] [reference-modifier]
```

4.3.8.3.3 Syntax Rules

(1) Argument-1 must be an identifier, a literal, or an arithmetic expression. Specific rules governing the number, class, and category of argument-1 are given in the definition of each function. (See page A-27, Intrinsic Function Module.)

(2) A reference-modifier may be specified only for functions of the category alphanumeric.

(3) A function-identifier which references an alphanumeric function may be specified anywhere in the general formats that an identifier is permitted and where the rules associated with the general formats do not specifically prohibit reference to functions, except as follows:

a. As a receiving operand of any statement.

b. Where the rules associated with the general formats require the data item being referenced to have particular characteristics (such as class and category, usage, size, sign, and permissible values) and the evaluation of the function according to its definition and the particular arguments specified would not have these characteristics.

(4) A function-identifier which references an integer or numeric function may be used only in an arithmetic expression.

4.3.8.3.4 General Rules

(1) The class and other characteristics of the function being referenced are determined by the function definition. (See page A-27, Intrinsic Functions.)

(2) At the time reference is made to a function, its arguments are evaluated individually in the order specified in the list of arguments, from left to right. An argument being evaluated may itself be a function-identifier or may be an expression containing function-identifiers. There is no restriction preventing the function referenced in evaluating an argument from being the same function as that for which the argument is specified.

IV-22 New paragraph 4.3.8.4, change the title to "Reference-Modifier".

IV-22 New paragraph 4.3.8.4.2, replace the general format with the following:

```
{data-name-1                                    }
{                                               } (leftmost-character-position:[length])
{FUNCTION function-name-1 [({argument-1} ... )]}
```

NOTE: Data-name-1 and FUNCTION function-name-1 (argument-1) are shown in the above format to provide context and are not part of the reference-modifier.

IV-22 New paragraph 4.3.8.4.3, add the following as new syntax rule 5:

(5) The function referenced by function-name-1 and its arguments, if any, must be an alphanumeric function.

IV-22 New paragraph 4.3.8.4.4, change the first sentence of general rule 1 to read: "Each character of a data item referenced by data-name-1 or by function-name-1 and its arguments, if any, is assigned an ordinal number incrementing by one from the leftmost position to the rightmost position."

IV-23 New paragraph 4.3.8.4.4, general rule 3a, add the following new last sentence: "If an ALL subscript is specified for an operand, the reference-modifier is applied to each of the implicitly specified elements of the table."

IV-23 New paragraph 4.3.8.4.4, general rule 3, add the following as new rule c:

c. If reference modification is specified in a function reference, the reference modification is evaluated immediately after evaluation of the function.

IV-23 New paragraph 4.3.8.4.4, general rule 4, change the first sentence to read: "Reference modification creates a unique data item that is a subset of the data item referenced by data-name-1 or by function-name-1 and its arguments, if any."

IV-23 New paragraph 4.3.8.4.4, general rule 4, paragraphs a and b, replace four occurrences of "the data item referenced by data-name-1" by "the data item referenced by data-name-1 or function-name-1 and its arguments, if any".

IV-23 New paragraph 4.3.8.4.4, general rule 5, second sentence, replace the word "It" by "When a function is referenced, the unique data item has the class and category of alphanumeric. When data-name-1 is specified, the unique data item".

IV-23 New paragraph 4.3.8.5, change to read:

4.3.8.5 Identifier

4.3.8.5.1 Function

An identifier is a syntactically correct sequence of character-strings and separators used to reference data uniquely.

When a data item other than a function is being referenced, identifier is a term used to reflect that a data-name, if not unique in a program, must be followed by a syntactically correct combination of qualifiers, subscripts, or reference modifiers necessary for uniqueness of reference. (See page X-4, Scope of Names.)

4.3.8.5.2 General Format

Format 1:

```
function-identifier-1
```

Format 2:

```
                                     ┌      ┌cd-name-1     ┐┐
               ┌┌IN┐             ┐   │ ┌IN┐ │              ││
data-name-1    ││  │ data-name-2 │...│ │  │ ┤file-name-1   ││
               └└OF┘             ┘   │ └OF┘ │              ││
                                     └      └report-name-1 ┘┘

     [({subscript} ... )] [reference-modifier]
```

4.3.8.5.3 Syntax Rules

(1) The words IN and OF are equivalent.

IV-45 Paragraph 8, insert the following before the list of reserved words:

The following is a list of COBOL reserved words for the seven required modules and the four optional modules of Report Writer, Communication, Debug, and Segmentation.

IV-46 Paragraph 8, insert the following after the list of reserved words:

The following is a list of COBOL reserved words for the optional Intrinsic Function module.

FUNCTION

V-3 (T) SYMBOLIC CHARACTERS clause, delete outermost set of braces.

V-7 (T) ORGANIZATION clause, delete second occurrence of a right bracket after the word SEQUENTIAL.

V-15 (T) Insert a terminal period following the last bracket in format 1.

V-15 (T) Delete the commas between data-name-1 through data-name-11, inclusively, in format 1.

V-16 (T) Insert a terminal period following the last bracket in format 3.

V-16 (T) Delete the commas between data-name-1 through data-name-6, inclusively, in format 3.

V-27 (T) PERFORM format, AFTER phrase, change "literal-3" immediately after the reserved word AFTER to "index-name-3".

V-28 (T) Insert "[END-REWRITE]" at the end of the first REWRITE statement.

V-38 Change the format for subscripting to read:

```
                      {ALL                                      }
                      {                                         }
                      {integer-1                                }
{condition-name-1}    {                  [{+}            ]      }
{                } ( {data-name-2       [{ } integer-2  ]      } ...)
{data-name-1     }    {                  [{-}            ]      }
                      {                  [{+}            ]      }
                      {index-name-1      [{ } integer-3  ]      }
                      {                  [{-}            ]      }
```

V-38 Add the following after the format for subscripting:

FUNCTION-IDENTIFIER:

```
FUNCTION function-name-1 [({argument-1} ... )] [reference-modifier]
```

V-38 Change the format for reference modification to read:

```
{data-name-1                                    }
{                                               } (leftmost-character-position:[length])
{FUNCTION function-name-1 [({argument-1} ... )] }
```

V-38 Change the format for identifier to read:

Format 1:

```
function-identifier-1
```

Format 2:

```
data-name-1 [{IN} data-name-2] ... [{IN} {cd-name-1     }]
            [{OF}                  {file-name-1   }]
                                   {report-name-1 }

     [({subscript} ... )] [reference-modifier]
```

V-38 Insert after page V-38 the contents of pages A-16 and A-17 which contain the general format for functions.

GENERAL FORMAT FOR INTRINSIC FUNCTIONS

```
FUNCTION ACOS (argument-1)

FUNCTION ANNUITY (argument-1 argument-2)

FUNCTION ASIN (argument-1)

FUNCTION ATAN (argument-1)

FUNCTION CHAR (argument-1)

FUNCTION COS (argument-1)

FUNCTION CURRENT-DATE

FUNCTION DATE-OF-INTEGER (argument-1)

FUNCTION DAY-OF-INTEGER (argument-1)

FUNCTION FACTORIAL (argument-1)

FUNCTION INTEGER (argument-1)

FUNCTION INTEGER-OF-DATE (argument-1)

FUNCTION INTEGER-OF-DAY (argument-1)

FUNCTION INTEGER-PART (argument-1)

FUNCTION LENGTH (argument-1)

FUNCTION LOG (argument-1)

FUNCTION LOG10 (argument-1)

FUNCTION LOWER-CASE (argument-1)

FUNCTION MAX ({argument-1} ... )

FUNCTION MEAN ({argument-1} ... )

FUNCTION MEDIAN ({argument-1} ... )

FUNCTION MIDRANGE ({argument-1} ... )

FUNCTION MIN ({argument-1} ... )

FUNCTION MOD (argument-1 argument-2)

FUNCTION NUMVAL (argument-1)

FUNCTION NUMVAL-C (argument-1 [argument-2])
```

GENERAL FORMAT FOR INTRINSIC FUNCTIONS

```
FUNCTION ORD (argument-1)

FUNCTION ORD-MAX ({argument-1} ... )

FUNCTION ORD-MIN ({argument-1} ... )

FUNCTION PRESENT-VALUE (argument-1 {argument-2} ...)

FUNCTION RANDOM [(argument-1)]

FUNCTION RANGE ({argument-1} ... )

FUNCTION REM (argument-1 argument-2)

FUNCTION REVERSE (argument-1)

FUNCTION SIN (argument-1)

FUNCTION SQRT (argument-1)

FUNCTION STANDARD-DEVIATION ({argument-1} ... )

FUNCTION SUM ({argument-1} ... )

FUNCTION TAN (argument-1)

FUNCTION UPPER-CASE (argument-1)

FUNCTION VARIANCE ({argument-1} ... )

FUNCTION WHEN-COMPILED
```

Page No.	Change to ANSI X3.23-1985 and ISO 1989-1985
VI-1	Paragraph 1.3, change title to read: "RESTRICTIONS ON OVERALL LANGUAGE".
VI-2	Renumber paragraphs 1.3.5 and 1.3.6 to 1.3.6 and 1.3.7, respectively; and add new paragraph 1.3.5 without boxing as follows: 1.3.5 Function-Identifier The availability of function-identifiers is dependent on whether the Intrinsic Function module is supported by the implementation.
VI-13 (T)	Paragraph 4.5.2, SYMBOLIC CHARACTERS clause, delete outermost set of braces.
VI-20 (T)	Paragraph 5.3.2 of the OCCURS clause, delete the box around the first occurrence of "[INDEXED BY {index-name-1} ...]".
VI-31 (T)	Paragraph 5.9.4 of the PICTURE clause, general rule 8, second line, change "explain" to "explained".
VI-50 (T)	Paragraph 5.15.6 of the VALUE clause, general rule 6, second line, change "or in a entry" to "or in an entry".
VI-57	Paragraph 6.3.1.2, replace the first paragraph after the general format beginning with "The usage of the operand..." by "Identifier-1 must reference a data item whose usage is explicitly or implicitly DISPLAY. If identifier-1 is a function-identifier, it must reference an alphanumeric function.".
VI-62	Paragraph 6.3.4, penultimate paragraph: a. Insert "(2)" at the beginning of the paragraph and indent 3 spaces. b. Change the first sentence in part to read: "Values are established for arithmetic expressions and functions if and when...".
VI-70	Paragraph 6.4.7, change in part to read: "...PICTURE clause or function definition, then the result...".
VI-96	Paragraph 6.18.4, general rule 4, change in part to read: "If any identifier is subscripted or is a function-identifier, the subscript or function-identifier is evaluated...".
VI-103	Paragraph 6.19.4, general rule 2, second paragraph, change the first sentence to read: "If identifier-1 is reference modified, subscripted, or is a function-identifier, the reference modifier, subscript, or function-identifier is evaluated only once, immediately before data is moved to the first of the receiving operands."
VI-104 (T)	Paragraph 6.19.4 of the MOVE statement, general rule 3, fourth line, change "alphabetic, numeric edited," to "alphabetic, alphanumeric, numeric edited,".
VI-105 (T)	Paragraph 6.19.4 of the MOVE statement, general rule 4c, change indentation to align with general rule 4b.

Page No.	Change to ANSI X3.23-1985 and ISO 1989-1985

VII-6 (T) Paragraph 2.1, in the Input-Output Section of the Sequential I-O module, delete the box around the general format:

```
[I-O-CONTROL.  [input-output-control-entry]]
```

VII-48 Paragraph 4.5.3, change syntax rule 1 to read:

(1) If identifier-1 is a function-identifier, it must reference an alphanumeric function. When identifier-1 is not a function-identifier, record-name-1 and identifier-1 must not reference the same storage area.

VII-52 Paragraph 4.7.3, change syntax rule 1 to read:

(1) If identifier-1 is a function-identifier, it must reference an alphanumeric function. When identifier-1 is not a function-identifier, record-name-1 and identifier-1 must not reference the same storage area.

VIII-30 Paragraph 4.6.3, change syntax rule 1 to read:

(1) If identifier-1 is a function-identifier, it must reference an alphanumeric function. When identifier-1 is not a function-identifier, record-name-1 and identifier-1 must not reference the same storage area.

VIII-30 (T) Paragraph 4.6.3, change syntax rule 4 in part to read: "...mode for which...".

VIII-37 Paragraph 4.9.3, change syntax rule 1 to read:

(1) If identifier-1 is a function-identifier, it must reference an alphanumeric function. When identifier-1 is not a function-identifier, record-name-1 and identifier-1 must not reference the same storage area.

IX-3 (T) Paragraph 1.3.4, entitled "I-O Status", second occurrence of a paragraph numbered (1), subparagraphs c and d, change indentation to align with subparagraph b.

IX-4 (T) Paragraph 1.3.4, entitled "I-O Status", second occurrence of a paragraph numbered (3), subparagraph b, first line, box "or rewrite".

IX-6 (T) Paragraph 1.3.5, entitled "The Invalid Key Condition", second occurrence of a paragraph numbered 2, first line, change "If not exception" to "If no exception".

IX-7 (T) Paragraph 1.3.7, first paragraph, last line, change "an" to "and".

IX-33 Paragraph 4.6.3, change syntax rule 1 to read:

(1) If identifier-1 is a function-identifier, it must reference an alphanumeric function. When identifier-1 is not a function-identifier, record-name-1 and identifier-1 must not reference the same storage area.

IX-33 (T) Paragraph 4.6.3 of the REWRITE statement in the Indexed I-O module, syntax rule 3, change to read: "The INVALID KEY phrase must be specified in the REWRITE statement for indexed files for which an appropriate USE AFTER STANDARD EXCEPTION procedure is not specified."

Page No.	Change to ANSI X3.23-1985 and ISO 1989-1985
IX-41	Paragraph 4.9.3, change syntax rule 1 to read: (1) If identifier-1 is a function-identifier, it must reference an alphanumeric function. When identifier-1 is not a function-identifier, record-name-1 and identifier-1 must not reference the same storage area.
X-1 (T)	Paragraph 1.1, Function for Inter-Program Communication module, fifth line, change "data value available" to "data values available".
X-2 (T)	Paragraph 1.3.4, entitled "External Objects and Internal Objects", second paragraph, last sentence, change "representative" to "representation".
X-5 (T)	Paragraph 1.3.8, entitled "Scope of Names", second paragraph on page X-5, second line, box the word "either". In the same paragraph, third and fourth lines, box "which contains a Configuration Section or in any program contained within that program".
X-6 (T)	Paragraph 1.3.8.1, entitled "Conventions for Program-Names", rule 3, box "except programs it directly or indirectly contains".
X-10 (T)	Paragraph 2.4.2, entitled "Programs in the Initial State", box numbered paragraphs 3 and 4.
X-19 (T)	Paragraph 4.3.1, Function for data description entry in the Inter-Program Communication module, first paragraph, last line, box "or global names".
X-19 (T)	Paragraph 4.3.1, Function for data description entry in the Inter-Program Communication module, second paragraph, second line, box "or external".
X-27	Paragraph 5.2.3, add new syntax rule 4 as follows: (4) Identifier-2 must not be a function-identifier.
X-29 (T)	Paragraph 5.2.4 of the CALL statement, general rule 10, change two occurrences of "data-names" to "parameters"; also change two occurrences of "data-name" to "parameter".
XI-8 (T)	Paragraph 4.1.3 of the MERGE statement, syntax rule 3, fifth line, change "in the file" to "in the files".
XI-13	Paragraph 4.2.3, change syntax rule 3 to read: (3) If identifier-1 is a function-identifier, it must reference an alphanumeric function. When identifier-1 is not a function-identifier, record-name-1 and identifier-1 must not reference the same storage area.
XII-4 (T)	Paragraph 2.4 of the COPY statement, general rule 7, fifth and sixth lines, change two occurrences of "pseudo-text-delimiter" to "pseudo-text delimiter".
XII-4 (T)	Paragraph 2.4 of the COPY statement, general rule 9, third paragraph, fourth line, change "When a text word" to "When a text word specified in the BY phrase is introduced, it appears on a debugging line if the first library text word being replaced is specified on a debugging line. Except".

XIII-7 (T) Paragraph 3.2.2 of the file description entry in the Report Writer module, VALUE OF clause, delete the fourth period in the ellipsis.

XIV-3 (T) Paragraph 2.2.2 of the communication description entry, delete the commas between data-name-1 through data-name-11, inclusively, in format 1.

XIV-4 (T) Paragraph 2.2.2 of the communication description entry, delete the commas between data-name-1 through data-name-6, inclusively, in format 3.

XIV-19 (T) Paragraph 3.2.4 of the DISABLE statement, general rule 4, third line, change "SOURCE)" to "SOURCE))".

XIV-26 Paragraph 3.6.3, add new syntax rule 6 as follows:

(6) If identifier-1 is a function-identifier, it must reference an alphanumeric function.

XV-5 (T) Paragraph 3.2.3 of the USE FOR DEBUGGING statement, syntax rule 10, last line, delete "or indexing".

XVII-8 (T) Paragraph 2.11, entitled "CODASYL COBOL JOURNAL OF DEVELOPMENT 1981", item 12, change "PROGRAM" to "PERFORM".

XVII-13 Add the following after paragraph 3.4:

3.5 AMERICAN NATIONAL STANDARD COBOL 1985, ADDENDUM 1

The X3J4 COBOL Technical Committee of the Accredited Standards Committee X3 was charged with the responsibility to develop addenda to American National Standard COBOL X3.23-1985 as a means of adding upward compatible language extensions. In June 1985, X3J4 began the task of preparing the first addendum. Language extensions considered for inclusion in Addendum 1 were taken from the CODASYL COBOL Journal of Development 1984.

In January 1987 X3J4 approved the content and format for the first draft proposed Addendum 1 to American National Standard COBOL X3.23-1985 and recommended to X3 that it be published for public review and comment. X3J4 held two public review and comment periods in which comments were received from the data processing community on the content of the draft proposed Addendum 1 to American National Standard COBOL X3.23-1985. X3J4 reviewed and responded to all comments received during these two public review periods.

In October 1988 X3J4 approved the final version of the draft proposed Addendum 1 to American National Standard COBOL X3.23-1985 and forwarded the document to the X3 committee for processing. The X3 committee then voted in favor of the acceptability of the draft proposed Addendum 1 to American National Standard COBOL X3.23-1985. This X3 vote also forwarded the proposed Addendum 1 for American National Standard COBOL X3.23-1985 to the American National Standards Institute for final approval.

Addendum 1 to American National Standard COBOL X3.23-1985 proposed by X3 was approved by the American National Standards Institute on September 13, 1989 as an

addendum to American National Standard COBOL X3.23-1985. The specifications of this approved Addendum 1 are published in the American National Standards Institute document X3.23A-1989.

XVII-15 Add the following after paragraph 4.3:

4.4 ISO STANDARD 1989-1985 FOR COBOL, ADDENDUM 1

At its meeting in Vienna, Austria, in February 1984, ISO Technical Committee 97, Subcommittee 22, Working Group 4 on COBOL resolved to propose addenda for upward compatible language extensions to ISO Standard 1989-1985 for COBOL. The purpose of proposing addenda of upward compatible language extensions instead of embarking immediately on a revision to Standard COBOL was to provide the language enhancements in a more timely manner, e.g., in a two to five year time frame rather than a five to ten year, or longer, time frame. At this same meeting ISO/TC97/SC22 Working Group 4 recommended that the United States be requested to provide draft documents for the addenda. The recommendations of ISO/TC97/SC22 Working Group 4 were approved and the work of developing the addenda was assigned to X3J4.

During X3J4's work on Addendum 1 for Standard COBOL, close and continuous liaison was maintained with the international community through ISO/IEC JTC1/SC22 Working Group 4. The draft document was presented for review and comment to ISO/IEC JTC1/SC22 in March 1987 as a draft proposed Addendum 1 to ISO Standard 1989-1985 for COBOL. ISO/IEC JTC1/SC22 unanimously approved a resolution to send the proposed Addendum 1 to ISO Standard 1989-1985 for COBOL to the Central Secretariat for registration as a draft international standard (DIS). The DIS Addendum 1 to ISO Standard 1989-1985 for COBOL was circulated to all the ISO member bodies for inquiry. Addendum 1 to ISO Standard 1989-1985 for COBOL was accepted by the ISO Council.

XVII-17 Add the following before the entry for "User-defined words"; align the word "Function-names" with the word "User-defined":

Function-names, system-names, and user-defined words form intersecting sets. . . . . N 1 ITR

XVII-18 Add the following before the entry for "Literals"; align the word "Function-names" one position to the right of the word "Literals":

Function-names . . . . . N 1 ITR

XVII-19 Change the entry for "Reference modification" to "Reference-modifier".

XVII-19 Add the following before the entry for "Reference-modifier"; align the word "Function-identifier" with the word "Reference-modifier":

Function-identifier . . . . . N 1 ITR

XVII-19 (T) Under "Reference Format", change entry in 3RD STD column for "Continuation of COBOL word, numeric literal" from "1 NUC" to "2 NUC".

XVII-31 Add the following before the entry for "ACCEPT statement":

Intrinsic functions	N	1 ITR
ACOS function	N	1 ITR
ANNUITY function	N	1 ITR
ASIN function	N	1 ITR
ATAN function	N	1 ITR
CHAR function	N	1 ITR
COS function	N	1 ITR
CURRENT-DATE function	N	1 ITR
DATE-OF-INTEGER function	N	1 ITR
DAY-OF-INTEGER function	N	1 ITR
FACTORIAL function	N	1 ITR
INTEGER function	N	1 ITR
INTEGER-OF-DATE function	N	1 ITR
INTEGER-OF-DAY function	N	1 ITR
INTEGER-PART function	N	1 ITR
LENGTH function	N	1 ITR
LOG function	N	1 ITR
LOG10 function	N	1 ITR
LOWER-CASE function	N	1 ITR
MAX function	N	1 ITR
MEAN function	N	1 ITR
MEDIAN function	N	1 ITR
MIDRANGE function	N	1 ITR
MIN function	N	1 ITR
MOD function	N	1 ITR
NUMVAL function	N	1 ITR
NUMVAL-C function	N	1 ITR
ORD function	N	1 ITR
ORD-MAX function	N	1 ITR
ORD-MIN function	N	1 ITR
PRESENT-VALUE function	N	1 ITR
RANDOM function	N	1 ITR
RANGE function	N	1 ITR
REM function	N	1 ITR
REVERSE function	N	1 ITR
SIN function	N	1 ITR
SQRT function	N	1 ITR
STANDARD-DEVIATION function	N	1 ITR
SUM function	N	1 ITR
TAN function	N	1 ITR
UPPER-CASE function	N	1 ITR
VARIANCE function	N	1 ITR
WHEN-COMPILED function	N	1 ITR

XVII-50 Add the following after item 107:

(108) Intrinsic Function Module (1 ITR). The Intrinsic Function module provides the capability to reference a data item whose value is derived automatically at the time of reference during the execution of the object program.

XVII-64 (T) Substantive change 26, entitled "PERFORM statement", prior to the last paragraph on the page, delete: "Under second Standard COBOL, PARA3 will be executed 8 times as shown above. Under third Standard COBOL, PARA3 will be executed 6 times as shown above."

XVII-70 (T) Substantive change 37, entitled "File position indicator", first paragraph, second line, change "access made" to "access mode".

XVII-71 Add the word FUNCTION to the list of reserved words in item 39.

XVII-93 Add the following after item 80:

(81) Integer and Numeric Functions. The implementor specifies the representation of integer and numeric functions, and this representation need not be the standard data format. (See 4.3.4, second occurrence of page number IV-16, on page A-10.)

(82) Types of Functions. The characteristics of the returned value are defined by the implementor. (See 2.3, paragraphs 2 and 3, on page A-29.)

(83) CHAR function. If the current program collating sequence was not specified by the ALPHABET clause, the implementor determines the value. (See 2.9.4, rule 2, on page A-37.)

(84) RANDOM function. If the first reference to this function in the run unit does not specify argument-1, the seed value is defined by the implementor. (See 2.35.3, rule 3, on page A-64.)

XVII-98 Add the following after item 28:

(29) Intrinsic functions. If, at the time a function is referenced, the arguments specified for that reference do not have values that comply with the specified constraints, the returned value for the function is undefined. (See 1.2.2, on page A-27.)

XVI-9 Insert after section XVI pages A-27 through A-76 containing the Intrinsic Function module.

XVIII-1 thru XVIII-13 Add the following entries to the Index in their appropriate alphabetical sequence:

ACOS function, A-33
Alphanumeric function, A-8, A-10, A-29
ALL subscript, A-9, A-11, A-28
ANNUITY function, A-34
Arccosine function, A-33
Arcsine function, A-35
Arctangent function, A-36
Argument, A-8, A-11, A-28
ASIN function, A-35
ATAN function, A-36
CHAR function, A-37
Character function, A-37
COS function A-38

SECTION A: INTRINSIC FUNCTION MODULE

1. INTRODUCTION TO THE INTRINSIC FUNCTION MODULE

1.1 PURPOSE OF INTRINSIC FUNCTION MODULE

The Intrinsic Function module provides the capability to reference a data item whose value is derived automatically at the time of reference during the execution of the object program.

1.2 LANGUAGE CONCEPTS

1.2.1 Function-Name

In the Intrinsic Function module, a function is a temporary data item whose value is determined by invoking a mechanism provided by the implementor at the time the function is referenced during the execution of a statement. A function-name names a mechanism provided by the implementor to determine the value of a function. A function-name is a COBOL word that is one of a specified list of COBOL words which may be used in COBOL source programs. (See page A-29, Definitions of Functions.)

1.2.2 Value Returned by a Function

The value returned by a function is considered to be a data value. A mechanism is provided at object time to assign a data value to a function when it is referenced. In order to determine the function's value, the evaluation mechanism may require access to data values provided by the referencing program. These data values are provided by specifying parameters, known as arguments, when referencing the function. Specific functions may place constraints on these arguments, such as range, etc. If, at the time a function is referenced, the arguments specified for that reference do not have values that comply with the specified constraints, the returned value for the function is undefined.

1.2.3 Function-Identifier

A function-identifier is used by the programmer to reference a function within the Procedure Division of a COBOL source program. (See page IV-22, Function-Identifier.)

2. GENERAL DESCRIPTION

2.1 FUNCTION DEFINITION AND RETURNED VALUE

The definition of a function identifies:

(1) For alphanumeric functions, the size of the returned value.

(2) For numeric and integer functions, the sign of the returned value and whether the function is integer.

(3) For some other cases, the value returned.

2.1.1 Date Conversion Function

The Gregorian calendar is used in the date conversion functions. The starting date of Monday, January 1, 1601, was chosen to establish a simple relationship between the Standard Date and DAY-OF-WEEK; i.e., integer date 1 was a Monday, DAY-OF-WEEK 1.

2.2 ARGUMENTS

Arguments specify values used in the evaluation of a function. Arguments are specified in the function-identifier. These arguments can be specified as identifiers, as arithmetic expressions, or as literals. The definition of a function specifies the number of arguments required, which can be zero, one, or more. For some functions, the number of arguments which can be specified may be variable. The order in which arguments are specified in a function-identifier determines the interpretation given to each value in arriving at the function value.

Arguments may be required to have a certain class or a subset of a certain class. The types of argument are:

(1) Numeric. An arithmetic expression must be specified. The value of the arithmetic expression, including operational sign, is used in determining the value of the function.

(2) Alphabetic. An elementary data item of the class alphabetic or a nonnumeric literal containing only alphabetic characters must be specified. The size associated with the argument can be used in determining the value of the function.

(3) Alphanumeric. A data item of the class alphabetic or alphanumeric or a nonnumeric literal must be specified. The size associated with the argument can be used in determining the value of the function.

(4) Integer. An arithmetic expression which will always result in an integer value must be specified. The value of the arithmetic expression, including operational sign, is used in determining the value of the function.

The rules for a function may place constraints on the permissible values for arguments in order to permit meaningful determination of the function's value. If, at the time a function is referenced, the arguments specified for that reference do not have values within the permissible range, the returned value for the function is undefined.

When the definition of a function permits an argument to be repeated a variable number of times, a table may be referenced by specifying the data-name and any qualifiers that identify the table, followed immediately by subscripting where one or more of the subscripts is the word ALL.

When ALL is specified as a subscript, the effect is as if each table element associated with that subscript position were specified. The order of the implicit specification of each occurrence is from left to right, with the first (or leftmost) specification being the identifier with each subscript specified by the word ALL replaced by one, the next specification being the same identifier with the rightmost subscript specified by the word ALL incremented by one. This process continues with the rightmost ALL subscript being incremented by one for each implicit specification until the rightmost ALL subscript has been incremented through its range of values. If there are any additional ALL subscripts, the ALL subscript immediately to the left of the rightmost ALL subscript is incremented by one, the rightmost ALL subscript is reset to one and the process of varying the rightmost ALL subscript is repeated. The ALL subscript to the left of the rightmost ALL subscript is incremented by one through its range of values. For each additional ALL subscript, this process is repeated in turn until the leftmost ALL subscript has been incremented by one through its range of values. If

the ALL subscript is associated with an OCCURS DEPENDING ON clause, the range of values is determined by the object of the OCCURS DEPENDING ON clause. The evaluation of an ALL subscript must result in at least one argument, otherwise the returned value is undefined.

2.3 TYPES OF FUNCTIONS

Data item functions are elementary data items and return alphanumeric, numeric, or integer values. Data item functions are treated as elementary data items and can not be receiving operands. Types of data item functions are:

(1) Alphanumeric functions. These are of the class and category alphanumeric. The number of character positions in this data item is specified in the function definition. Alphanumeric functions have an implicit usage of DISPLAY.

(2) Numeric functions. These are of the class and category numeric. A numeric function is always considered to have an operational sign. Those characteristics of the returned value not otherwise specified for a given function are defined by the implementor.

A numeric function may be used only in an arithmetic expression.

A numeric function may not be referenced where an integer operand is required, even though a particular reference may yield an integer value.

(3) Integer functions. These are of the class and category numeric. An integer function is always considered to have an operational sign. Those characteristics of the returned value not otherwise specified for a given function are defined by the implementor.

An integer function may be used only in an arithmetic expression.

An integer function can be referenced where an integer operand is required and where a signed operand is allowed.

2.4 DEFINITIONS OF FUNCTIONS

Table 1 on pages A-30 through A-32 summarizes the functions that are available.

The Arguments column defines the type and number of arguments as follows:

A means alphabetic
I means integer
N means numeric
X means alphanumeric

The Type column defines the type of the function as follows:

I means integer
N means numeric
X means alphanumeric

FUNCTION-NAME	ARGUMENTS	TYPE	VALUE RETURNED
ACOS	N1	N	Arccosine of N1
ANNUITY	N1, I2	N	Ratio of annuity paid for I2 periods at interest of N1 to initial investment of one
ASIN	N1	N	Arcsine of N1
ATAN	N1	N	Arctangent of N1
CHAR	I1	X	Character in position I1 of program collating sequence
COS	N1	N	Cosine of N1
CURRENT-DATE	None	X	Current date and time and difference from Greenwich Mean Time
DATE-OF-INTEGER	I1	I	Standard date equivalent (YYYYMMDD) of integer date
DAY-OF-INTEGER	I1	I	Julian date equivalent (YYYYDDD) of integer date
FACTORIAL	I1	I	Factorial of I1
INTEGER	N1	I	The greatest integer not greater than N1
INTEGER-OF-DATE	I1	I	Integer date equivalent of standard date (YYYYMMDD)
INTEGER-OF-DAY	I1	I	Integer date equivalent of Julian date (YYYYDDD)
INTEGER-PART	N1	I	Integer part of N1
LENGTH	A1 or N1 or X1	I	Length of argument
LOG	N1	N	Natural logarithm of N1
LOG10	N1	N	Logarithm to base 10 of N1
LOWER-CASE	A1 or X1	X	All letters in the argument are set to lowercase

Table 1: Table of Functions

FUNCTION-NAME	ARGUMENTS	TYPE	VALUE RETURNED
MAX	A1 ... or I1 ... or N1 ... or X1 ...	Depends upon arguments*	Value of maximum argument
MEAN	N1 ...	N	Arithmetic mean of arguments
MEDIAN	N1 ...	N	Median of arguments
MIDRANGE	N1 ...	N	Mean of minimum and maximum arguments
MIN	A1 ... or I1 ... or N1 ... or X1 ...	Depends upon arguments*	Value of minimum argument
MOD	I1, I2	I	I1 modulo I2
NUMVAL	X1	N	Numeric value of simple numeric string
NUMVAL-C	X1, X2	N	Numeric value of numeric string with optional commas and currency sign
ORD	A1 or X1	I	Ordinal position of the argument in collating sequence
ORD-MAX	A1 ... or N1 ... or X1 ...	I	Ordinal position of maximum argument
ORD-MIN	A1 ... or N1 ... or X1 ... or	I	Ordinal position of minimum argument
PRESENT-VALUE	N1 N2 ...	N	Present value of a series of future period-end amounts, N2, at a discount rate of N1
RANDOM	I1	N	Random number
RANGE	I1 ... or N1 ...	Depends upon argument	Value of maximum argument minus value of minimum argument
REM	N1, N2	N	Remainder of N1/N2

*A function that has only alphabetic arguments is type alphanumeric.

Table 1: Table of Functions (Continued)

FUNCTION-NAME	ARGUMENTS	TYPE	VALUE RETURNED
REVERSE	A1 or X1	X	Reverse order of the characters of the argument
SIN	N1	N	Sine of N1
SQRT	N1	N	Square root of N1
STANDARD-DEVIATION	N1 ...	N	Standard deviation of arguments
SUM	I1 ... or N1 ...	Depends upon arguments	Sum of arguments
TAN	N1	N	Tangent of N1
UPPER-CASE	A1 or X1	X	All letters in the argument are set to uppercase
VARIANCE	N1 ...	N	Variance of argument
WHEN-COMPILED	None	X	Date and time program was compiled

Table 1: Table of Functions (Continued)

2.5 THE ACOS FUNCTION

2.5.1 Description

The ACOS function returns a numeric value in radians that approximates the arccosine of argument-1. The type of this function is numeric.

2.5.2 General Format

```
FUNCTION ACOS (argument-1)
```

2.5.3 Arguments

(1) Argument-1 must be class numeric.

(2) The value of argument-1 must be greater than or equal to -1 and less than or equal to +1.

2.5.4 Returned Values

(1) The returned value is the approximation of the arccosine of argument-1 and is greater than or equal to zero and less than or equal to pi.

2.6 THE ANNUITY FUNCTION

2.6.1 Description

The ANNUITY function (annuity immediate) returns a numeric value that approximates the ratio of an annuity paid at the end of each period for the number of periods specified by argument-2 to an initial investment of one. Interest is earned at the rate specified by argument-1 and is applied at the end of the period, before the payment. The type of this function is numeric.

2.6.2 General Format

```
FUNCTION ANNUITY (argument-1 argument-2)
```

2.6.3 Arguments

(1) Argument-1 must be class numeric.

(2) The value of argument-1 must be greater than or equal to zero.

(3) Argument-2 must be a positive integer.

2.6.4 Returned Values

(1) When the value of argument-1 is zero, the value of the function is the approximation of:

1 / argument-2

(2) When the value of argument-1 is not zero, the value of the function is the approximation of:

argument-1 / (1 - (1 + argument-1) ** (- argument-2))

2.7 THE ASIN FUNCTION

2.7.1 Description

The ASIN function returns a numeric value in radians that approximates the arcsine of argument-1. The type of this function is numeric.

2.7.2 General Format

```
FUNCTION ASIN (argument-1)
```

2.7.3 Arguments

(1) Argument-1 must be class numeric.

(2) The value of argument-1 must be greater than or equal to -1 and less than or equal to +1.

2.7.4 Returned Values

(1) The returned value is the approximation of the arcsine of argument-1 and is greater than or equal to -pi/2 and less than or equal to +pi/2.

2.8 THE ATAN FUNCTION

2.8.1 Description

The ATAN function returns a numeric value in radians that approximates the arctangent of argument-1. The type of this function is numeric.

2.8.2 General Format

```
FUNCTION ATAN (argument-1)
```

2.8.3 Arguments

(1) Argument-1 must be class numeric.

2.8.4 Returned Values

(1) The returned value is the approximation of the arctangent of argument-1 and is greater than -pi/2 and less than +pi/2.

2.9 THE CHAR FUNCTION

2.9.1 Description

The CHAR function returns a one-character alphanumeric value that is a character in the program collating sequence having the ordinal position equal to the value of argument-1. The type of this function is alphanumeric.

2.9.2 General Format

```
FUNCTION CHAR (argument-1)
```

2.9.3 Arguments

(1) Argument-1 must be an integer.

(2) The value of argument-1 must be greater than zero and less than or equal to the number of positions in the collating sequence.

2.9.4 Returned Values

(1) If more than one character has the same position in the program collating sequence, the character returned as the function value is that of the first literal specified for that character position in the ALPHABET clause.

(2) If the current program collating sequence was not specified by an ALPHABET clause, the implementor determines the value.

2.10 THE COS FUNCTION

2.10.1 Description

The COS function returns a numeric value that approximates the cosine of an angle or arc, expressed in radians, that is specified by argument-1. The type of this function is numeric.

2.10.2 General Format

```
FUNCTION COS (argument-1)
```

2.10.3 Arguments

(1) Argument-1 must be class numeric.

2.10.4 Returned Values

(1) The returned value is the approximation of the cosine of argument-1 and is greater than or equal to -1 and less than or equal to +1.

2.11 THE CURRENT-DATE FUNCTION

2.11.1 Description

The CURRENT-DATE function returns a 21-character alphanumeric value that represents the calendar date, time of day, and local time differential factor provided by the system on which the function is evaluated. The type of this function is alphanumeric.

2.11.2 General Format

```
FUNCTION CURRENT-DATE
```

2.11.3 Returned Values

(1) The character positions returned, numbered from left to right, are:

Character Positions	Contents
1-4	Four numeric digits of the year in the Gregorian calendar.
5-6	Two numeric digits of the month of the year, in the range 01 through 12.
7-8	Two numeric digits of the day of the month, in the range 01 through 31.
9-10	Two numeric digits of the hours past midnight, in the range 00 through 23.
11-12	Two numeric digits of the minutes past the hour, in the range 00 through 59.
13-14	Two numeric digits of the seconds past the minute, in the range 00 through 59.
15-16	Two numeric digits of the hundredths of a second past the second, in the range 00 through 99. The value 00 is returned if the system on which the function is evaluated does not have the facility to provide the fractional part of a second.
17	Either the character '-', the character '+', or the character '0'. The character '-' is returned if the local time indicated in the previous character positions is behind Greenwich Mean Time. The character '+' is returned if the local time indicated is the same as or ahead of Greenwich Mean Time. The character '0' is returned if the system on which this function is evaluated does not have the facility to provide the local time differential factor.
18-19	If character position 17 is '-', two numeric digits are returned in the range 00 through 12 indicating the number of hours that the reported time is behind Greenwich Mean Time. If character position 17 is '+', two numeric digits are returned in the range 00 through 13 indicating the number of hours that the reported time is ahead of Greenwich Mean Time. If character position 17 is '0', the value 00 is returned.

Character Positions	Contents
20-21	Two numeric digits are returned in the range 00 through 59 indicating the number of additional minutes that the reported time is ahead of or behind Greenwich Mean Time, depending on whether character position 17 is '+' or '-', respectively. If character position 17 is '0', the value 00 is returned.

(2) If the system does not have the facility to provide fractional parts of a second, the value 00 is returned in character positions 15 and 16.

(3) If the system does not have the facility to provide the local time differential factor, the value 00000 is returned in character positions 17 through 21.

2.12 THE DATE-OF-INTEGER FUNCTION

2.12.1 Description

The DATE-OF-INTEGER function converts a date in the Gregorian calendar from integer date form to standard date form (YYYYMMDD). The type of this function is integer.

2.12.2 General Format

```
FUNCTION DATE-OF-INTEGER (argument-1)
```

2.12.3 Arguments

(1) Argument-1 is a positive integer that represents a number of days succeeding December 31, 1600, in the Gregorian calendar.

2.12.4 Returned Values

(1) The returned value represents the ISO Standard date equivalent of the integer specified in argument-1.

(2) The returned value is in the form (YYYYMMDD) where YYYY represents a year in the Gregorian calendar; MM represents the month of that year; and DD represents the day of that month.

2.13 THE DAY-OF-INTEGER FUNCTION

2.13.1 Description

The DAY-OF-INTEGER function converts a date in the Gregorian calendar from integer date form to Julian date form (YYYYDDD). The type of this function is integer.

2.13.2 General Format

```
FUNCTION DAY-OF-INTEGER (argument-1)
```

2.13.3 Arguments

(1) Argument-1 is a positive integer that represents a number of days succeeding December 31, 1600, in the Gregorian calendar.

2.13.4 Returned Values

(1) The returned value represents the Julian equivalent of the integer specified in argument-1.

(2) The returned value is an integer of the form (YYYYDDD) where YYYY represents a year in the Gregorian calendar and DDD represents the day of that year.

2.14 THE FACTORIAL FUNCTION

2.14.1 Description

The FACTORIAL function returns an integer that is the factorial of argument-1. The type of this function is integer.

2.14.2 General Format

```
FUNCTION FACTORIAL (argument-1)
```

2.14.3 Arguments

(1) Argument-1 must be an integer greater than or equal to zero.

2.14.4 Returned Values

(1) If the value of argument-1 is zero, the value 1 is returned.

(2) If the value of argument-1 is positive, its factorial is returned.

2.15 THE INTEGER FUNCTION

2.15.1 Description

The INTEGER function returns the greatest integer value that is less than or equal to the argument. The type of this function is integer.

2.15.2 General Format

```
FUNCTION INTEGER (argument-1)
```

2.15.3 Arguments

(1) Argument-1 must be class numeric.

2.15.4 Returned Values

(1) The returned value is the greatest integer less than or equal to the value of argument-1. For example, if the value of argument-1 is -1.5, -2 is returned. If the value of argument-1 is +1.5, +1 is returned.

2.16 THE INTEGER-OF-DATE FUNCTION

2.16.1 Description

The INTEGER-OF-DATE function converts a date in the Gregorian calendar from standard date form (YYYYMMDD) to integer date form. The type of this function is integer.

2.16.2 General Format

```
FUNCTION INTEGER-OF-DATE (argument-1)
```

2.16.3 Arguments

(1) Argument-1 must be an integer of the form YYYYMMDD, whose value is obtained from the calculation (YYYY * 10,000) + (MM * 100) + DD.

a. YYYY represents the year in the Gregorian calendar. It must be an integer greater than 1600.

b. MM represents a month and must be a positive integer less than 13.

c. DD represents a day and must be a positive integer less than 32 provided that it is valid for the specified month and year combination.

2.16.4 Returned Values

(1) The returned value is an integer that is the number of days the date represented by argument-1 succeeds December 31, 1600, in the Gregorian calendar.

2.17 THE INTEGER-OF-DAY FUNCTION

2.17.1 Description

The INTEGER-OF-DAY function converts a date in the Gregorian calendar from Julian date form (YYYYDDD) to integer date form. The type of this function is integer.

2.17.2 General Format

```
FUNCTION INTEGER-OF-DAY (argument-1)
```

2.17.3 Arguments

(1) Argument-1 must be an integer of the form YYYYDDD, whose value is obtained from the calculation (YYYY * 1000) + DDD.

a. YYYY represents the year in the Gregorian calendar. It must be an integer greater than 1600.

b. DDD represents the day of the year. It must be a positive integer less than 367 provided that it is valid for the year specified.

2.17.4 Returned Values

(1) The returned value is an integer that is the number of days the date represented by argument-1 succeeds December 31, 1600, in the Gregorian calendar.

2.18 THE INTEGER-PART FUNCTION

2.18.1 Description

The INTEGER-PART function returns an integer that is the integer portion of argument-1. The type of this function is integer.

2.18.2 General Format

```
FUNCTION INTEGER-PART (argument-1)
```

2.18.3 Arguments

(1) Argument-1 must be class numeric.

2.18.4 Returned Values

(1) If the value of argument-1 is zero, the returned value is zero.

(2) If the value of argument-1 is positive, the returned value is the greatest integer less than or equal to the value of argument-1. For example, if the value of argument-1 is +1.5, +1 is returned.

(3) If the value of argument-1 is negative, the returned value is the least integer greater than or equal to the value of argument-1. For example, if the value of argument-1 is -1.5, -1 is returned.

2.19 THE LENGTH FUNCTION

2.19.1 Description

The LENGTH function returns an integer equal to the length of the argument in character positions. The type of this function is integer.

2.19.2 General Format

```
FUNCTION LENGTH (argument-1)
```

2.19.3 Arguments

(1) Argument-1 may be a nonnumeric literal or a data item of any class or category.

(2) If argument-1 or any data item subordinate to argument-1 is described with the DEPENDING phrase of the OCCURS clause, the contents of the data item referenced by the data-name specified in the DEPENDING phrase are used at the time the LENGTH function is evaluated.

2.19.4 Returned Values

(1) If argument-1 is a nonnumeric literal or an elementary data item or argument-1 is a group data item that does not contain a variable occurrence data item, the value returned is an integer equal to the length of argument-1 in character positions.

(2) If argument-1 is a group data item containing a variable occurrence data item, the returned value is an integer determined by evaluation of the data item specified in the DEPENDING phrase of the OCCURS clause for that variable occurrence data item. This evaluation is accomplished according to the rules in the OCCURS clause dealing with the data item as a sending data item. (See page VI-26, The OCCURS Clause; and page VI-46, The USAGE Clause.)

(3) The returned value includes implicit FILLER characters, if any.

2.20 THE LOG FUNCTION

2.20.1 Description

The LOG function returns a numeric value that approximates the logarithm to the base e (natural log) of argument-1. The type of this function is numeric.

2.20.2 General Format

```
FUNCTION LOG (argument-1)
```

2.20.3 Arguments

(1) Argument-1 must be class numeric.

(2) The value of argument-1 must be greater than zero.

2.20.4 Returned Values

(1) The returned value is the approximation of the logarithm to the base e of argument-1.

2.21 THE LOG10 FUNCTION

2.21.1 Description

The LOG10 function returns a numeric value that approximates the logarithm to the base 10 of argument-1. The type of this function is numeric.

2.21.2 General Format

```
FUNCTION LOG10 (argument-1)
```

2.21.3 Arguments

(1) Argument-1 must be class numeric.

(2) The value of argument-1 must be greater than zero.

2.21.4 Returned Values

(1) The returned value is the approximation of the logarithm to the base 10 of argument-1.

2.22 THE LOWER-CASE FUNCTION

2.22.1 Description

The LOWER-CASE function returns a character string that is the same length as argument-1 with each upppercase letter replaced by the corresponding lowercase letter. The type of this function is alphanumeric.

2.22.2 General Format

```
FUNCTION LOWER-CASE (argument-1)
```

2.22.3 Arguments

(1) Argument-1 must be class alphabetic or alphanumeric and must be at least one character in length.

2.22.4 Returned Values

(1) The same character string as argument-1 is returned, except that each uppercase letter replaced by the corresponding lowercase letter.

(2) The character string returned has the same length as argument-1.

(3) If the computer character set does not include lowercase letters, no changes take place in the character string.

2.23 THE MAX FUNCTION

2.23.1 Description

The MAX function returns the content of the argument-1 that contains the maximum value. The type of this function depends upon the argument types as follows:

Argument Type	Function Type
Alphabetic	Alphanumeric
Alphanumeric	Alphanumeric
All arguments integer	Integer
Numeric (some arguments may be integer)	Numeric

2.23.2 General Format

```
FUNCTION MAX ({argument-1} ... )
```

2.23.3 Arguments

(1) If more than one argument-1 is specified, all arguments must be of the same class.

2.23.4 Returned Values

(1) The returned value is the content of the argument-1 having the greatest value. The comparisons used to determine the greatest value are made according to the rules for simple conditions. (See page VI-54, Simple Conditions.)

(2) If more than one argument-1 has the same greatest value, the content of the argument-1 returned is the leftmost argument-1 having that value.

(3) If the type of the function is alphanumeric, the size of the returned value is the same as the size of the selected argument-1.

2.24 THE MEAN FUNCTION

2.24.1 Description

The MEAN function returns a numeric value that is the arithmetic mean (average) of its arguments. The type of this function is numeric.

2.24.2 General Format

```
FUNCTION MEAN ({argument-1} ... )
```

2.24.3 Arguments

(1) Argument-1 must be class numeric.

2.24.4 Returned Values

(1) The returned value is the arithmetic mean of the argument-1 series.

(2) The returned value is defined as the sum of the argument-1 series divided by the number of occurrences referenced by argument-1.

2.25 THE MEDIAN FUNCTION

2.25.1 Description

The MEDIAN function returns the content of the argument whose value is the middle value in the list formed by arranging the arguments in sorted order. The type of this function is numeric.

2.25.2 General Format

```
FUNCTION MEDIAN ({argument-1} ... )
```

2.25.3 Arguments

(1) Argument-1 must be class numeric.

2.25.4 Returned Values

(1) The returned value is the content of the argument-1 having the middle value in the list formed by arranging all the argument-1 values in sorted order.

(2) If the number of occurrences referenced by argument-1 is odd, the returned value is such that at least half of the occurrences referenced by argument-1 are greater than or equal to the returned value and at least half are less than or equal. If the number of occurrences referenced by argument-1 is even, the returned value is the arithmetic mean of the values referenced by the two middle occurrences.

(3) The comparisons used to arrange the argument-1 values in sorted order are made according to the rules for simple conditions. (See page VI-54, Simple Conditions.)

2.26 THE MIDRANGE FUNCTION

2.26.1 Description

The MIDRANGE (middle range) function returns a numeric value that is the arithmetic mean (average) of the values of the minimum argument and the maximum argument. The type of this function is numeric.

2.26.2 General Format

```
FUNCTION MIDRANGE ({argument-1} ... )
```

2.26.3 Arguments

(1) Argument-1 must be class numeric.

2.26.4 Returned Values

(1) The returned value is the arithmetic mean of the greatest argument-1 value and the least argument-1 value. The comparisons used to determine the greatest and least values are made according to the rules for simple conditions. (See page VI-54, Simple Conditions.)

2.27 THE MIN FUNCTION

2.27.1 Description

The MIN function returns the content of the argument-1 that contains the minimum value. The type of this function depends upon the argument types as follows:

Argument Type	Function Type
Alphabetic	Alphanumeric
Alphanumeric	Alphanumeric
All arguments integer	Integer
Numeric (some arguments may be integer)	Numeric

2.27.2 General Format

```
FUNCTION MIN ({argument-1} ... )
```

2.27.3 Arguments

(1) If more than one argument-1 is specified, all arguments must be of the same class.

2.27.4 Returned Values

(1) The returned value is the content of the argument-1 having the least value. The comparisons used to determine the least value are made according to the rules for simple conditions. (See page VI-54, Simple Conditions.)

(2) If more than one argument-1 has the same least value, the content of the argument-1 returned is the leftmost argument-1 having that value.

(3) If the type of the function is alphanumeric, the size of the returned value is the same as the size of the selected argument-1.

2.28 THE MOD FUNCTION

2.28.1 Description

The MOD function returns an integer value that is argument-1 modulo argument-2. The type of this function is integer.

2.28.2 General Format

```
FUNCTION MOD (argument-1 argument-2)
```

2.28.3 Arguments

(1) Argument-1 and argument-2 must be integers.

(2) The value of argument-2 must not be zero.

2.28.4 Returned Values

(1) The returned value is argument-1 modulo argument-2. The returned value is defined as:

argument-1 - (argument-2 * FUNCTION INTEGER (argument-1 / argument-2))

(2) The following illustrates the expected results for some values of argument-1 and argument-2.

Argument-1	Argument-2	Return
11	5	1
-11	5	4
11	-5	-4
-11	-5	-1

2.29 THE NUMVAL FUNCTION

2.29.1 Description

The NUMVAL function returns the numeric value represented by the character string specified by argument-1. Leading and trailing spaces are ignored. The type of this function is numeric.

2.29.2 General Format

```
FUNCTION NUMVAL (argument-1)
```

2.29.3 Arguments

(1) Argument-1 must be a nonnumeric literal or alphanumeric data item whose content has one of the following two formats:

```
[space] [+] [space] {digit [ . [digit]]} [space]
        [-]         {. digit           }
```

or

```
[space] {digit [ . [digit]]} [space] [ + ] [space]
        {. digit           }         [ - ]
                                     [ CR ]
                                     [ DB ]
```

where space is a string of zero or more spaces and digit is a string of one to 18 digits.

(2) The total number of digits in argument-1 must not exceed 18.

(3) If the DECIMAL-POINT IS COMMA clause is specified in the SPECIAL-NAMES paragraph, a comma must be used in argument-1 rather than a decimal point.

2.29.4 Returned Values

(1) The returned value is the numeric value represented by argument-1.

(2) The number of digits returned is 18.

2.30 THE NUMVAL-C FUNCTION

2.30.1 Description

The NUMVAL-C function returns the numeric value represented by the character string specified by argument-1. Any optional currency sign specified by argument-2 and any optional commas preceding the decimal point are ignored. The type of this function is numeric.

2.30.2 General Format

```
FUNCTION NUMVAL-C (argument-1 [argument-2])
```

2.30.3 Arguments

(1) Argument-1 must be a nonnumeric literal or alphanumeric data item whose content has one of the following two formats:

```
[space] [+] [space] [cs] [space] {digit [, digit] ... [. [digit]]} [space]
        [-]                      {. digit                        }
```

or

```
[space] [cs] [space] {digit [, digit] ... [. [digit]]} [space] [+ ] [space]
                     {. digit                        }         [- ]
                                                               [CR]
                                                               [DB]
```

where space is a string of zero or more spaces, cs is the string of one or more characters specified by argument-2 and digit is a string of one or more digits.

(2) If the DECIMAL-POINT IS COMMA clause is specified in the SPECIAL-NAMES paragraph, the functions of the comma and decimal point in argument-1 are reversed.

(3) The total number of digits in argument-1 must not exceed 18.

(4) Argument-2, if specified, must be a nonnumeric literal or alphanumeric data item.

(5) If argument-2 is not specified, the character used for cs is the currency symbol specified for the program.

2.30.4 Returned Values

(1) The returned value is the numeric value represented by argument-1.

(2) The number of digits returned is 18.

2.31 THE ORD FUNCTION

2.31.1 Description

The ORD function returns an integer value that is the ordinal position of argument-1 in the collating sequence for the program. The lowest ordinal position is 1. The type of this function is integer.

2.31.2 General Format

```
FUNCTION ORD (argument-1)
```

2.31.3 Arguments

(1) Argument-1 must be one character in length and must be class alphabetic or alphanumeric.

2.31.4 Returned Values

(1) The returned value is the ordinal position of argument-1 in the collating sequence for the program.

2.32 THE ORD-MAX FUNCTION

2.32.1 Description

The ORD-MAX function returns a value that is the ordinal number of the argument-1 that contains the maximum value. The type of this function is integer.

2.32.2 General Format

```
FUNCTION ORD-MAX ({argument-1} ... )
```

2.32.3 Arguments

(1) If more than one argument-1 is specified, all arguments must be of the same class.

2.32.4 Returned Values

(1) The returned value is the ordinal number that corresponds to the position of the argument-1 having the greatest value in the argument-1 series.

(2) The comparisons used to determine the greatest valued argument are made according to the rules for simple conditions. (See page VI-54, Simple Conditions.)

(3) If more than one argument-1 has the same greatest value, the number returned corresponds to the position of the leftmost argument-1 having that value.

2.33 THE ORD-MIN FUNCTION

2.33.1 Description

The ORD-MIN function returns a value that is the ordinal number of the argument that contains the minimum value. The type of this function is integer.

2.33.2 General Format

```
FUNCTION ORD-MIN ({argument-1} ... )
```

2.33.3 Arguments

(1) If more than one argument-1 is specified, all arguments must be of the same class.

2.33.4 Returned Values

(1) The returned value is the ordinal number that corresponds to the position of the argument-1 having the least value in the argument-1 series.

(2) The comparisons used to determine the least valued argument-1 are made according to the rules for simple conditions. (See page VI-54, Simple Conditions.)

(3) If more than one argument-1 has the same least value, the number returned corresponds to the position of the leftmost argument-1 having that value.

2.34 THE PRESENT-VALUE FUNCTION

2.34.1 Description

The PRESENT-VALUE function returns a value that approximates the present value of a series of future period-end amounts specified by argument-2 at a discount rate specified by argument-1. The type of this function is numeric.

2.34.2 General Format

```
FUNCTION PRESENT-VALUE (argument-1 {argument-2} ... )
```

2.34.3 Arguments

(1) Argument-1 and argument-2 must be of the class numeric.

(2) The value of argument-1 must be greater than -1.

2.34.4 Returned Values

(1) The returned value is an approximation of the summation of a series of calculations with each term in the following form:

argument-2 / (1 + argument-1) ** n

There is one term for each occurrence of argument-2. The exponent, n, is incremented from one by one for each term in the series.

2.35 THE RANDOM FUNCTION

2.35.1 Description

The RANDOM function returns a numeric value that is a pseudo-random number from a rectangular distribution. The type of this function is numeric.

2.35.2 General Format

```
FUNCTION RANDOM [(argument-1)]
```

2.35.3 Arguments

(1) If argument-1 is specified, it must be zero or a positive integer. It is used as the seed value to generate a sequence of pseudo-random numbers.

(2) If a subsequent reference specifies argument-1, a new sequence of pseudo-random numbers is started.

(3) If the first reference to this function in the run unit does not specify argument-1, the seed value is defined by the implementor.

(4) In each case, subsequent references without specifying argument-1 return the next number in the current sequence.

2.35.4 Returned Values

(1) The returned value is greater than or equal to zero and less than one.

(2) For a given seed value on a given implementation, the sequence of pseudo-random numbers will always be the same.

(3) The implementor will specify the subset of the domain of argument-1 values that will yield distinct sequences of pseudo-random numbers. This subset must include the values from 0 through at least 32767.

2.36 THE RANGE FUNCTION

2.36.1 Description

The RANGE function returns a value that is equal to the value of the maximum argument minus the value of the minimum argument. The type of this function depends upon the argument types as follows:

Argument Type	Function Type
All arguments integer	Integer
Numeric (some arguments may be integer)	Numeric

2.36.2 General Format

```
FUNCTION RANGE ({argument-1} ... )
```

2.36.3 Arguments

(1) Argument-1 must be class numeric.

2.36.4 Returned Values

(1) The returned value is equal to the greatest value of argument-1 minus the least value of argument-1. The comparisons used to determine the greatest and least values are made according to the rules for simple conditions. (See page VI-54, Simple Conditions.)

2.37 THE REM FUNCTION

2.37.1 Description

The REM function returns a numeric value that is the remainder of argument-1 divided by argument-2. The type of this function is numeric.

2.37.2 General Format

```
FUNCTION REM (argument-1 argument-2)
```

2.37.3 Arguments

(1) Argument-1 and argument-2 must be class numeric.

(2) The value of argument-2 must not be zero.

2.37.4 Returned Values

(1) The returned value is the remainder of argument-1 / argument-2. It is defined as the expression:

```
argument-1 - (argument-2 * FUNCTION INTEGER-PART (argument-1 / argument-2))
```

2.38 THE REVERSE FUNCTION

2.38.1 Description

The REVERSE function returns a character string of exactly the same length as argument-1 and whose characters are exactly the same as those of argument-1, except that they are in reverse order. The type of this function is alphanumeric.

2.38.2 General Format

```
FUNCTION REVERSE (argument-1)
```

2.38.3 Arguments

(1) Argument-1 must be class alphabetic or alphanumeric and must be at least one character in length.

2.38.4 Returned Values

(1) If argument-1 is a character string of length n, the returned value is a character string of length n such that for $1 \leq j \leq n$, the character in position j of the returned value is the character from position n-j+1 of argument-1.

2.39 THE SIN FUNCTION

2.39.1 Description

The SIN function returns a numeric value that approximates the sine of an angle or arc, expressed in radians, that is specified by argument-1. The type of this function is numeric.

2.39.2 General Format

```
FUNCTION SIN (argument-1)
```

2.39.3 Arguments

(1) Argument-1 must be class numeric.

2.39.4 Returned Values

(1) The returned value is the approximation of the sine of argument-1 and is greater than or equal to -1 and less than or equal to +1.

2.40 THE SQRT FUNCTION

2.40.1 Description

The SQRT function returns a numeric value that approximates the square root of argument-1. The type of this function is numeric.

2.40.2 General Format

```
FUNCTION SQRT (argument-1)
```

2.40.3 Arguments

(1) Argument-1 must be class numeric.

(2) The value of argument-1 must be zero or positive.

2.40.4 Returned Values

(1) The returned value is the absolute value of the approximation of the square root of argument-1.

2.41 THE STANDARD-DEVIATION FUNCTION

2.41.1 Description

The STANDARD-DEVIATION function returns a numeric value that approximates the standard deviation of its arguments. The type of this function is numeric.

2.41.2 General Format

```
FUNCTION STANDARD-DEVIATION ({argument-1} ... )
```

2.41.3 Arguments

(1) Argument-1 must be class numeric.

2.41.4 Returned Values

(1) The returned value is the approximation of the standard deviation of the argument-1 series.

(2) The returned value is calculated as follows:

a. The difference between each argument-1 value and the arithmetic mean of the argument-1 series is calculated and squared.

b. The values obtained are then added together. This quantity is divided by the number of values in the argument-1 series.

c. The square root of the quotient obtained is then calculated. The returned value is the absolute value of this square root.

(3) If the argument-1 series consists of only one value, or if the argument-1 series consists of all variable occurrence data items and the total number of occurrences for all of them is one, the returned value is zero.

2.42 THE SUM FUNCTION

2.42.1 Description

The SUM function returns a value that is the sum of the arguments. The type of this function depends upon the argument types as follows:

Argument Type	Function Type
All arguments integer	Integer
Numeric (some arguments may be integer)	Numeric

2.42.2 General Format

```
FUNCTION SUM ({argument-1} ... )
```

2.42.3 Arguments

(1) Argument-1 must be class numeric.

2.42.4 Returned Values

(1) The returned value is the sum of the arguments.

(2) If the argument-1 series are all integers, the value returned is an integer.

(3) If the argument-1 series are not all integers, a numeric value is returned.

2.43 THE TAN FUNCTION

2.43.1 Description

The TAN function returns a numeric value that approximates the tangent of an angle or arc, expressed in radians, that is specified by argument-1. The type of this function is numeric.

2.43.2 General Format

```
FUNCTION TAN (argument-1)
```

2.43.3 Arguments

(1) Argument-1 must be class numeric.

2.43.4 Returned Values

(1) The returned value is the approximation of the tangent of argument-1.

2.44 THE UPPER-CASE FUNCTION

2.44.1 Description

The UPPER-CASE function returns a character string that is the same length as argument-1 with each lowercase letter replaced by the corresponding uppercase letter. The type of this function is alphanumeric.

2.44.2 General Format

```
FUNCTION UPPER-CASE (argument-1)
```

2.44.3 Arguments

(1) Argument-1 must be class alphabetic or alphanumeric and must be at least one character in length.

2.44.4 Returned Values

(1) The same character string as argument-1 is returned, except that each lowercase letter is replaced by the corresponding uppercase letter.

(2) The character string returned has the same length as argument-1.

2.45 THE VARIANCE FUNCTION

2.45.1 Description

The VARIANCE function returns a numeric value that approximates the variance of its arguments. The type of this function is numeric.

2.45.2 General Format

```
FUNCTION VARIANCE ({argument-1} ... )
```

2.45.3 Arguments

(1) Argument-1 must be class numeric.

2.45.4 Returned Values

(1) The returned value is the approximation of the variance of the argument-1 series.

(2) The returned value is defined as the square of the standard deviation of the argument-1 series. (See page A-70, STANDARD-DEVIATION Returned Values, rule 2.)

(3) If the argument-1 series consists of only one value, or if the argument-1 series consists of all variable occurrence data items and the total number of occurrences for all of them is one, the returned value is zero.

2.46 THE WHEN-COMPILED FUNCTION

2.46.1 Description

The WHEN-COMPILED function returns the date and time the program was compiled as provided by the system on which the program was compiled. The type of this function is alphanumeric.

2.46.2 General Format

```
FUNCTION WHEN-COMPILED
```

2.46.3 Returned Values

(1) The character positions returned, numbered from left to right, are:

Positions	Contents
1-4	Four numeric digits of the year in the Gregorian calendar.
5-6	Two numeric digits of the month of the year, in the range 01 through 12.
7-8	Two numeric digits of the day of the month, in the range 01 through 31.
9-10	Two numeric digits of the hours past midnight, in the range 00 through 23.
11-12	Two numeric digits of the minutes past the hour, in the range 00 through 59.
13-14	Two numeric digits of the seconds past the minute, in the range 00 through 59.
15-16	Two numeric digits of the hundredths of a second past the second, in the range 00 through 99. The value 00 is returned if the system on which the program was compiled did not have the facility to provide the fractional part of a second.
17	Either the character '-', the character '+', or the character '0'. The character '-' is returned if the local time of compilation, reported in the previous character positions, is behind Greenwich Mean Time. The character '+' is returned if the reported time is the same as or ahead of Greenwich Mean Time. The character '0' is returned if the system on which the program was compiled did not have the facility to provide the local time differential factor.
18-19	If character position 17 is '-', two numeric digits are returned in the range 00 through 12 indicating the number of hours that the reported time is behind Greenwich Mean Time. If character position 17 is '+', two numeric digits are returned in the range 00 through 13 indicating the number of hours that the reported time is ahead of Greenwich Mean Time. If character position 17 is '0', the value 00 is returned.

Character Positions	Contents
20-21	Two numeric digits are returned in the range 00 through 59 indicating the number of additional minutes that the reported time is ahead of or behind Greenwich Mean Time, depending on whether character position 17 is '+' or '-', respectively. If character position 17 is '0', the value 00 is returned.

(2) The returned value is the date and time of compilation of the source program that contains this function. If the program is a contained program, the returned value is the compilation date and time associated with the separately compiled program in which it is contained.

(3) The returned value must denote the same time as the compilation date and time if provided in the listing of the source program and in the generated object code for the source program, although their representations and precisions may differ.